Gestión de múltiples impresoras 3D y detección de fallos a través de IA.

David Muñoz Bermudo

Resum– Este proyecto pretende presentar una solución para mejorar la accesibilidad y control de múltiples impresoras 3D, aspirando así a ahorrar tiempo y dedicación para obtener los resultados esperados. Además, se pretende incorporar una herramienta de visión por computador para monitorizar y detectar posibles fallos en la impresión para que, en caso de falla, se ahorre material y se prevenga una posible avería de la máquina. La solución que este proyecto presenta está especialmente enfocada a la gestión y uso de una sala de impresión 3D en un centro educativo, donde el profesor o encargado no tenga que dedicar o perder tiempo extra para utilizarlas sino que se facilite su uso al máximo para potenciar su uso.

Paraules clau– Impresión 3D, visión por computador, gestión y optimizacion, centros educativos.

Abstract– This project aims to present a solution to improve the accessibility and control of multiple 3D printers, thus aiming to save time and dedication to obtain the expected results. In addition, it is intended to incorporate a computer vision tool to monitor and detect possible printing failures preventing plastic wasting or a possible breakdown of the machine. The solution that this project presents is especially focused on the management and use of a 3D printing room in an school, where the teacher or person in charge does not have to spend or waste extra time to use them but to facilitate their use to the maximum to enhance their use.

Keywords– 3D print, computer vision, management and optimization, schools.

1 INTRODUCCIÓ

1.1. Antecedentes

La tecnología de la impresión 3D, aunque sus orígenes se remontan a la década de los 90, no ha sido hasta bien entrado el nuevo siglo que se empezó a popularizar y extender fuera del ámbito de investigación para pasar a formar parte de las industrias, FabLabs, uso personal y más adelante, en los centros educativos. Dentro de la impresión 3D hay muchos tipos de máquinas que utilizan diferentes técnicas las más convencionales y más comunes son las que utilizan la técnica llamada "Fused Deposition Modeling" o FMD, que se basan en ir depositando un material, normalmente un polímero termofusible en una superficie plana e ir superponiendo capas unas encima de otra hasta conseguir el modelo 3D deseado. Detrás de esto hay muchas tecnologías, tanto de hardware como software, desde las partes mecánicas y electrónicas de la máquina, pasando por el firmware, hasta los programas necesarios para pasar de un archivo 3d a un archivo entendible por la máquina. Estas impresoras se han convertido en un instrumento ideal para los FabLabs para la creación de piezas o partes de sus productos o para desarrollar nuevas ideas. Y en el sector educativo es una herramienta que tiene mucho potencial tanto porque permite crear cualquier tipo de proyectos relacionados con la tecnología, como para proporcionar todo tipo de material docente como maquetas, ejemplos prácticos, experimentos, etc.

1.2. Motivación

Actualmente la mayoría de sala de impresoras 3d dedicadas a un uso docente o en un FabLab no consta de ningún sistema que se encargue de optimizar y mejorar el control y accesiblidad de estas.
Existen muchas herramientas independientes para hacer algunas mejoras pero nadie usa un sistema que las integre para poder sacarle el máximo rendimiento a este tipo de salas. Además, esta tecnología es muy útil en ámbitos donde no se dispone necesariamente de una persona experta en estas

• E-mail de contacte: dmbdavid97@gmail.com
• Mención realitzada: Computació
• Treball tutoritzat per: Jordi Carrabina Aróztegui y Marc Codina Barbera
• Curs 2020/21

máquinas, que aún así deberían poder utilizarse de la manera más eficiente posible.
En sectores como el de la docencia por ejemplo, donde a nivel educativo la impresión 3D aporta una gran variedad de soluciones y usos, no se está consiguiendo una buena implementación puesto que la mayoría de máquinas de bajo coste que hay en el mercado requieren de un tiempo de aprendizaje y dedicación elevado del que el profesor o el personal de soporte normalmente no dispone o no quiere invertir. Y aunque hay centros que disponen de impresoras, al no existir ningún sistema integrado de gestión, monitorización y control, frecuentemente se producen fallos que redundan en una pérdida de tiempo y material y en un progresivo abandono de la tecnología.

Este proyecto propone una solución a estos problemas, y pretende desarrollar una herramienta que permita controlar las múltiples impresoras disponibles en, por ejemplo, un colegio y desde donde se pueda monitorizar la impresión en curso, añadir nuevas impresiones y detectar fallos de impresión que la paren y generen un aviso, mediante modelos de visión por computador, se detectara el supuesto fallo y lanzará una señal para suspender la impresión en curso.

Además, para la implementación de este sistema se quiere prescindir de utilizar PCs dedicados a ninguna de las partes del sistema, por motivos de seguridad y precio, ya que los sistemas empotrados son más eficientes para este propósito, requieren menos recursos y mantenimiento, y evitan el riesgo de ser reutilizados para otras tareas. Complementariamente, se pretende independizar lo máximo posible, el sistema de monitorización de la red de comunicaciones del lugar en el que esté instalado, para evitar que se pueda acceder al sistema interno de la impresora, haciendo que el mismo dispositivo que aloje el portal web de la herramienta, actúe como firewall y como nodo gestor de la red local.

1.3. Objetivos

El objetivo del proyecto es la realización de una solución basada en sistemas empotrados para gestionar múltiples impresoras 3D que integre una solución de visión por computador que detecte fallos de impresión.

Para alcanzarlo dividiremos el problema en los siguientes subobjetivos.

- **Proponer e implementar la infraestructura del proyecto.** Se requerirá de varios dispositivos: (1) uno empotrado para alojar el portal web, el servicio de visión por computador y (2) otros empotrados de menores prestaciones para el control de cada una de las impresoras; (3) una webcam para cada impresora (con la correspondiente iluminación) para la adquisición de video de la impresión para su posterior análisis; y (4) la configuración de la red privada de las impresoras y su conexión exterior (incluyendo firewall, etc.).

- **Desarrollar una aplicación web de gestión integrada.** Esta tendrá que comunicarse con las diferentes APIs del sistema para: (1) controlar las opciones básicas de las impresoras (encender/apagar, imprimir archivos, datos de impresión, etc.); (2) mostrar el vídeo en tiempo real de las impresiones disponibles; y (3) gestionar el detector de fallos de impresión y generar los eventos correspondientes (paro, aviso, etc.)

- **Desarrollar un sistema de visión por computador portable.** Compuesto por varias webcam, sus iluminaciones, un empotrado con prestaciones suficientes para gestionar un número determinado de impresiones y el SW correspondiente. Validarlo en diferentes configuraciones y analizar su rendimiento.

- **Desarrollar un HW a añadir a la impresora** que permita controlar la iluminación, la webcam y la impresora mandando las órdenes correspondientes por puerto serie sobre USB.

- **Analizar el funcionamiento del sistema y validar su rango de aplicación** para conseguir optimizar sus funciones.

2 ELECCIÓN DE LOS COMPONENTES DE LA SOLUCIÓN

2.1. Componentes SW

Existen varias soluciones para mejorar la accesibilidad y las capacidades de control de impresoras 3D actualmente, pero están centradas en un problema concreto. Lo que buscamos en este proyecto es aprovechar esas tecnologías que ya existen y montar un sistema que englobe la mayoría de ellas para aprovechar su potencial de forma centralizada. A continuación, se presenta un listado de las tecnologías que existen y que usaremos para integrar nuestro sistema.

- **Klipper [5]:** Firmware para impresoras 3D que busca mejorar las prestaciones de la mayoría de firmwares clásicos como marlin [6]. La gran diferencia es que, además de ser un firmware que se carga dentro del controlador de la impresora, también es un servicio que se ejecuta en un sistema linux (como una raspberry [10]). Busca aprovechar la capacidad de cómputo de este para mejorar la calidad de las operaciones mediante cálculos más complejos para realizar ajustes en, por ejemplo, la aceleración, el flujo y la velocidad de la máquina, logrando así una mejor impresión. El controlador interno de la impresora ejecuta la acción que recibe por puerto serie de la raspberry que es quien procesa el fichero a imprimir.

- **Octoprint [7]:** Aplicación de control de impresoras 3D de código abierto en forma de un servicio que se ejecuta dentro de la raspberry pi conectada a la impresora por usb. Consta de un portal web desde donde acceder a las funcionalidades de la impresora, además de una gran variedad de plugins hechos por la comunidad. Entre ellos uno para klipper. Con octoprint y klipper instalados, la comunicación se realiza con un firmware que se comunica con el hardware final. Esto aporta funcionalidades como el acceso simultáneo, desde la interfaz de la impresora (pantalla LCD), a las mismas funcionalidades que desde el portal de octoprint. El objetivo es usar este software junto con el plugin como puente entre nuestro portal web y la impresora en sí.

- **TheSpaghettiDetective [2]:** Un software open source que se constituye de un servidor web y un modelo entrenado específicamente para la detección de "hilos" de material de impresión mediante imágenes. Estos hilos se producen normalmente cuando una impresión falla. Para utilizarlo se requiere tener Octoprint y enviar las imágenes a través de un plugin desde donde esté alojado octoprint hasta el servidor donde se procesarán las imágenes. Mediante la interfaz web de este software se puede acceder y configurar múltiples impresoras pero las opciones que brinda son pobres. Se pretende instalar este software en el mismo servidor que aloja el portal web y acceder a la variable que mide la "calidad" de la impresión, por medio de su API; y mostrar los datos en nuestra interfaz, así ejecutar acciones como abortar la impresión cuando alcance un umbral concreto.

- **Django [8]:** Django es un framework de aplicaciones web de código abierto escrito en Python. Hemos elegido esta tecnología para desarrollar nuestro portal web.

- **Docker[9]:** Docker es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos. Docker utiliza características de aislamiento de recursos del kernel Linux, tales como cgroups y espacios de nombres (namespaces) para permitir que contenedores independientes se ejecuten dentro de una sola instancia de Linux, evitando la sobrecarga de iniciar y mantener máquinas virtuales. Utilizaremos esta tecnología para desplegar tanto nuestro servidor web como el servidor de visión por computador en un sistema encastado de forma sencilla, sin tener que preocuparnos de dependencias ni problemas de compatibilidad.

2.2. Componentes HW

A cotninuación una lista de los componentes hardware principales que se usara para el desarrollo del proyecto:

- **Impresoras 3D:** Usaremos las impresoras de Creality, modelo Ender 3.

- **Raspberry Pi 3b:** Alojará el software que se encargará de controlar la impresora con un cable USB y tendrá conectada por USB una webcam encargada de recoger imágenes para monitorizar la impresión

- **Webcam:** Hemos elegido la webcam 720p mas economica que hemos encontrado, y la logitech C720 para comparar cuanto afecta la calidad de la camara.

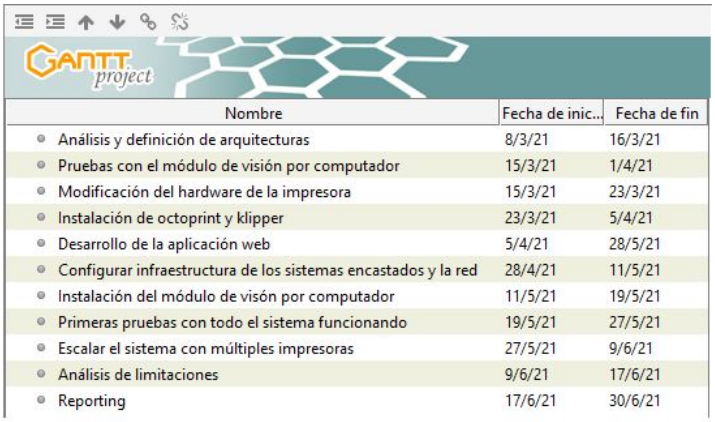
- **Iluminación:** Tira LED de 12V.

- **Nvidia JETSON Nano: [1]** Es una plataforma desarrollada por Nvidia para inferencia en modelos de inteligencia artificial. Es aquí donde desplegamos el servidor de visión por computador y el portal web de gestión, cada uno dentro de un contenedor virtual pero dentro de la misma red local para que puedan comunicarse.

- **Gateway/router:** Dongle WIFI USB conectado a la jetson para generar un hotspot.

3 METODOLOGÍA

3.1. Planificación



Nombre	Fecha de inic...	Fecha de fin
Análisis y definición de arquitecturas	8/3/21	16/3/21
Pruebas con el módulo de visión por computador	15/3/21	1/4/21
Modificación del hardware de la impresora	15/3/21	23/3/21
Instalación de octoprint y klipper	23/3/21	5/4/21
Desarrollo de la aplicación web	5/4/21	28/5/21
Configurar infraestructura de los sistemas encastados y la red	28/4/21	11/5/21
Instalación del módulo de visón por computador	11/5/21	19/5/21
Primeras pruebas con todo el sistema funcionando	19/5/21	27/5/21
Escalar el sistema con múltiples impresoras	27/5/21	9/6/21
Análisis de limitaciones	9/6/21	17/6/21
Reporting	17/6/21	30/6/21

Fig. 1: Lista de tareas a realizar durante el proyecto ordenadas cronológicamente. Creado con la herramienta Gantt-project [3]

Para la realización de este proyecto en las fechas indicadas hemos dividido el trabajo en una serie de tareas a realizar y les hemos asignado una fecha en función de la duración estimada que hemos asumido. Algunas de estas tareas requieren de otras para poder realizarse y otras se pueden ejecutar en paralelo. Es posible que a lo largo del desarrollo del proyecto estas fechas o duraciones varían así que nos valdremos de un diagrama de gantt que iremos actualizando para tener un buen control del desarrollo de los acontecimientos. En la figura 1 podemos ver las tareas identificadas así como sus fechas iniciales y finales, y en el A.1 podemos encontrar tanto el diagrama de gantt previsto como el ejecutado.

A continuación una pequeña descripción de cada una de las fases a realizar:

1. Estudiar en profundidad el problema para poder diseñar una arquitectura concreta para el sistema que vamos a implementar.

2. Instalar y probar el módulo de detección de fallos de impresión en un sistema a parte para ver su rendimiento y evaluar si nos conviene utilizarlo.

3. Diseñar e implementar el circuito electrónico que nos permita apagar y encender la máquina a través de nuestro controlador encastado, así como un led que instalaremos para tener iluminada siempre la zona de impresión.

4. Instalar en la raspberry Octoprint así como el pluggin para el firmware que utilizaremos, klipper, y flashearlo al controlador interno de la máquina.

5. Desarrollar un aplicativo web para centralizar múltiples impresoras en una misma interfaz así como mostrar el estado de la detección de error que viene dado del módulo de visión por computador.

6. Desplegar los diferentes dispositivos encastados e interconectarlos según la distribución de red que hayamos diseñado previamente.
7. Instalar el módulo de visión por computador dentro de nuestro servidor principal, donde compartirá recursos con nuestra aplicación web.
8. Trazar una serie de pruebas para realizar con el sistema esté correctamente desplegado y analizar su funcionamiento.
9. Añadir múltiples impresoras al sistema, en concreto dos más para tener un total de 3 y realizar nuevamente pruebas para medir el rendimiento según la escalabilidad del sistema así como su capacidad.
10. Dado el resultado de todas las pruebas realizadas anteriormente realizar un estudio en profundidad de las limitaciones de todo el sistema, tanto de la interfaz web como del módulo de visión por computador.
11. Sintetizar todo lo realizado en documentos, entre ellos uno en formato póster y otro en formato paper.

3.2. Herramientas

Para tener un buen control del desarrollo del proyecto utilizaremos la herramienta gratuita trello [4] para administrar el proyecto, a través de un tablero con columnas y items. Utilizaremos esto tanto para el desarrollo del proyecto general, como para el desarrollo de la aplicación web.

Para la organización del código utilizaremos github y crearemos un repositorio del portal web para llevar un buen control así como para que sea más accesible a la hora de desplegarlo en el sistema final, ya que podremos clonar el repositorio dentro del sistema encastado y ejecutar el instalador.

El IDE que utilizare para desarrollar la aplicación será Visual Code, dado que es muy versátil y ofrece una gran cantidad de ventajas al tener integrado componentes externos como git o docker que es la herramienta ideal para crear contenedores, lo usaremos para desplegar todo nuestro sistema.

Para acceder al sistema linux de los diferentes dispositivos encastados utilizaremos la herramienta putty [13], que nos permite crear conexiones ssh en windows de forma sencilla, además dado que nos conviene poder ejecutar aplicaciones gráficas y no solo la terminal, para dado el caso, poder acceder a la configuración del portal web antes de tener el sistema de red bien configurado. Para ello tendremos que habilitar el X11 forwarding así como instalar el servidor Xming [12] que es una implementación del sistema de ventanas X de unix para windows.

4 Análisis

A continuación analizaremos las partes del sistema y cómo se organizan en nuestra solución.

4.1. Arquitectura del Sistema

Las diferentes raspberries estarán conectadas mediante Wifi a una red privada de la jetson, haciendo esta última de hotspot Wifi previamente configurado por medio de una interfaz wifi USB, dado que en la Jetson no viene integrado. Por otro lado estará conectada a la red general de donde se va a desplegar el sistema mediante un cable ethernet, para que, sea accesible a todos los dispositivos conectados a la red.

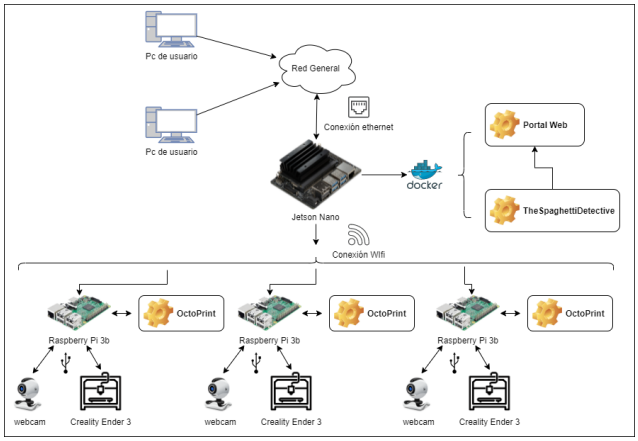


Fig. 2: Diagrama del sistema

4.2. Distribución de la red

Queremos que el sistema sea seguro y que no sea posible acceder a la red interna de las impresoras. Para ello, la separamos de la red general desde donde solo será accesible el servidor que contiene nuestra herramienta, la jetson. Además, privamos de acceso a internet a esta para evitar posibles conexiones desde el exterior.

Para que un usuario conectado a la red general, sea capaz de llegar a la raspberry y así acceder a su webcam, necesitaremos asignar un puerto de la Jetson para cada raspberry, de esta manera será posible acceder a las raspberrys de la red interna pero solo a través de la jetson, así se crea un mejor control.

También hay que tener en cuenta que los contenedores que contendrán el portal web y el módulo de visión por computador deben estar en la red local de la jetson y no en una red interna. Se deberá configurar a la hora de desplegar los contenedores con docker.

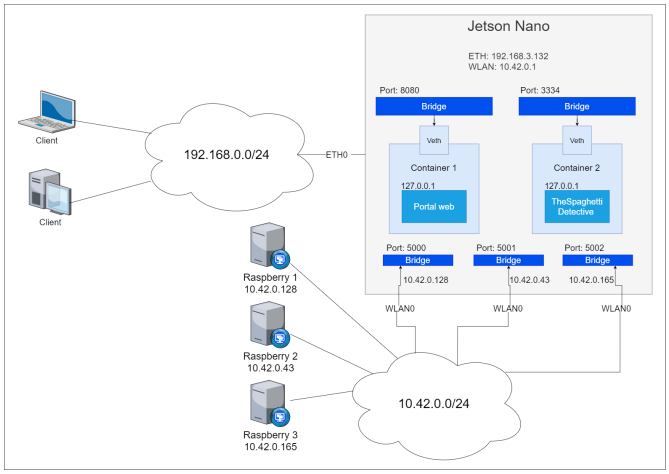


Fig. 3: Diagrama de la distribución de la red

4.3. Sistema de video

Para monitorizar la impresión en curso y automiatizar la parada en caso de fallo tendremos una webcam conectada a la raspberry de cada impresora captando video continuamente. Este estará subido a un servidor de video utilizando MJPG desde donde la jetson accedera para analizar dicho video y mandar la señal de parada en caso de que sea necesario, así como el portal accedera al video para mostrarlo en el navegador para el usuario. Las propiedades del video de entrada que utilizamos son 640x480 y 10 fps, dado que, como han mostrado los tests, es una buena configuración.

5 Implementación

5.1. Mecánica

Necesitaremos una pequeña carcasa donde montar el HW propio. La siguiente figura muestra la pieza inferior de la caja diseñada a medida e impresa en una de nuestras impresoras 3D.

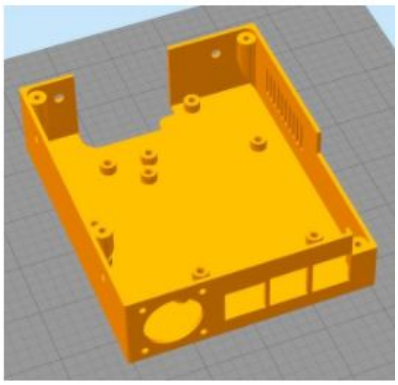


Fig. 4: Carcasa raspberry diseñada.

5.2. Hardware propio

Una vez tenemos claro el sistema que hay que montar el primer paso es tener preparado los componentes hardware, esto quiere decir diseñar un sistema eléctrico que utiliza la misma fuente de alimentación de la impresora para alimentar una raspberry y esta conectarla a relés para controlar el encendido de la propia máquina como el de una luz.

La impresora tiene dos pines de alimentación libres podemos sacar la tensión necesaria para alimentar el hardware propio, mediante un regulador de tensión ya que pasa de 24V a 12V para la iluminación y 5V para la raspberry. El esquema se muestra en el A.2. A continuación se muestra una imagen del sistema final.

Fig. 5: Hardware montado y en funcionamiento.

5.3. Software

Inicialmente hemos probado la API de octoprint, la cual controla la impresora, así como la comunicación con el módulo de IA, al que, una vez validado, se accede mediante peticiones http.

Posteriormente pasamos a crear el proyecto de django y codificar todos los modelos y vistas para nuestra herramienta.

El flujo de control desde que interaccionan con la herramienta hasta que se ejecuta una orden en la máquina es el siguiente: cada botón o cada elemento visual con el que se puede interactuar hace una llamada ajax con javascript, la cual va a buscar el modelo correspondiente a la acción ejecutada, se ejecuta el modelo devolviendo los datos necesarios a javascript para actualizar la página si es necesario. De esta manera, en toda la página de control tenemos datos que se actualizan sin necesidad de recargar la página así como controlar acciones.

Una vez desarrolladas todas las funcionalidades de nuestra aplicación, hemos tenido que buscar la mejor manera para desplegarla en producción dentro de nuestro dispositivo encantado, jetson nano. Para ello hemos utilizado la tecnología de contenedores Docker, que nos asegura un correcto despliegue en el dispositivo final compatible con seguir desarrollando en un entorno local. También hemos tenido que montar un servidor nginx para redireccionar las peticiones a django así como servir ficheros estáticos o manejar subidas de archivos.

Al final tenemos dos dockerfiles, uno para el servidor, y otro para la aplicación django y un docker-compose que se encarga de realizar todas las acciones necesarias para que estos elementos funcionen correctamente. Así simplemente para poner el servidor en producción, solo se tiene que clonar el repositorio de github y ejecutar un comando.

5.4. Sistema

Ahora tenemos todos los elementos listos para interconectarse. Así que una vez tenemos la aplicación web y el módulo de IA ejecutándose vamos a configurar la red para su uso. Primero creamos un hotspot wifi en la Jetson gracias a un dongle wifi usb externo, y conectamos las raspberrys de cada máquina a esa red.

Acto seguido configuramos tanto la herramienta como el módulo de IA con la IP privada de cada raspberry. Cabe destacar que para comunicarse con el software interno de las raspberries (octoprint) se utiliza un sistema seguro, que respecto al portal web funciona mediante API key, y la herramienta de IA se loguea a través de un plugin que hace de intermediario.

Listo esto solo queda configurar la red para poder acceder a las rasberrys de forma individual desde una red general, no la del hotspot wifi ya que esa será la interna exclusiva para la comunicación raspberry-jetson.

Para ello creamos reglas de routing A.3 para redireccionar unos puertos de la jetson a cada ip de la red interna que corresponde a cada máquina. A continuación prohibimos el acceso a internet a la jetson y de esta manera queda el sistema totalmente blindado.

Como último detalle redireccionamos el puerto de nuestro portal web, 8080, al puerto 80 para poder acceder sin

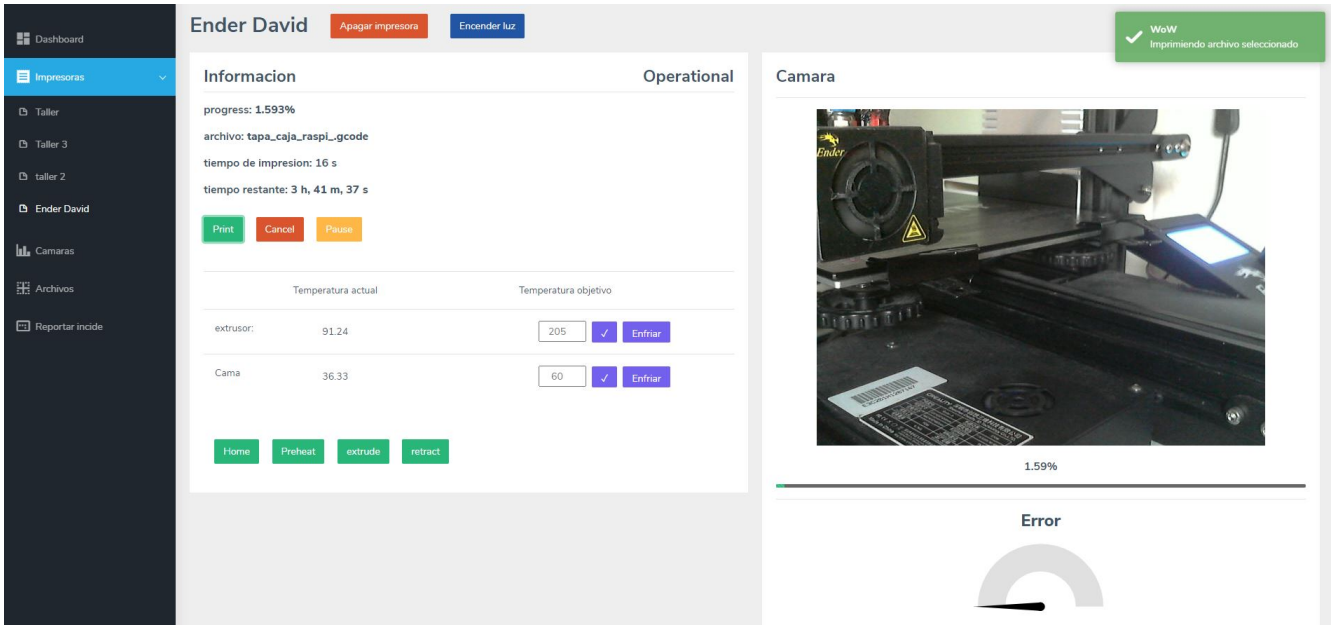


Fig. 6: Herramienta de gestión

necesidad de introducir el puerto.

6 TESTS Y RESULTADOS

6.1. Tests del módulo de IA

Lo primero que probaremos será el funcionamiento de la detección de fallos mediante video. Para ello instalamos el software TheSpaghettiDetective en un pc y realizamos una serie de pruebas simples. En las figuras 7, 8, y 9 podemos ver los archivos a imprimir para probar el módulo.

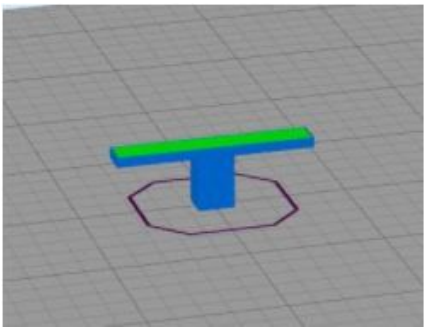


Fig. 7: modelo 3d del test 1

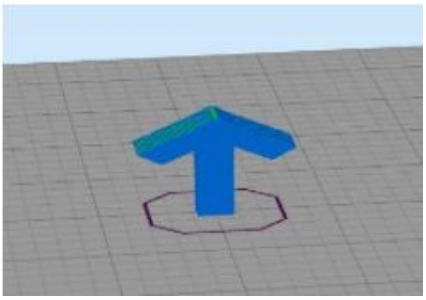


Fig. 8: modelo 3d del test 2

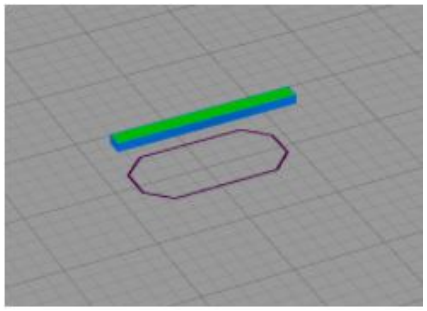


Fig. 9: modelo 3d del test 3

Test nº1

La primera prueba contiene ángulos de 90 grados para forzar a la máquina a que deje alguna imperfección que TSD pueda detectar.

La segunda directamente forzamos a la impresora a que imprima partes en el aire.

Y por último hacemos que directamente empiece a imprimir en el aire sin una base enganchada a la cama.

A continuación mostramos los resultados de los diferentes test respectivamente:

	test 1 v1	test 2 v1	test 3 V1
link	link	link	link
max error	0/12	12/12	5/12

Por cada frame del video (640x480 y 10 fps) devuelve un valor entre 0 y 12 cuánto la impresión está fallando, como indicación de la cantidad de hilos que ha ido detectando a lo largo del video.

Resultados:

En el primer test en el que hay “rastro de espaguetis”, estos se aprecian poco en el video y por tanto el SW no consigue detectar ninguno, probablemente dado a la mala calidad de la imagen. El segundo y tercer test muestran buenos resultados de detección a pesar de la calidad,

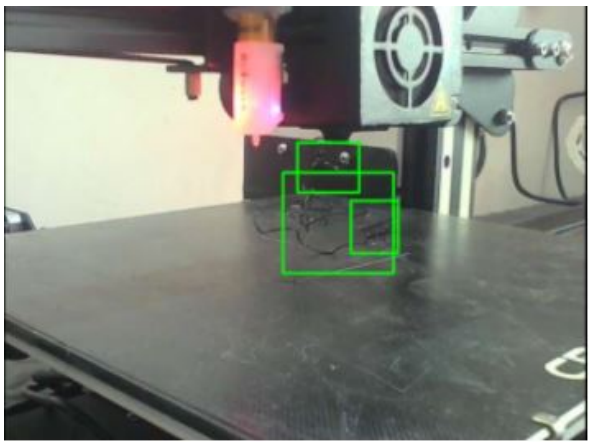


Fig. 10: Ejemplo de detección de hilos

gracias también a que contenían mayor número de fallos. El último test no ha llegado devolver un valor de error alto a pesar de ser un claro ejemplo de impresión fallida dado a que el tiempo de la impresión ha sido muy corto, apenas 5 minutos. He podido observar también que detecta mejor las partes que están más iluminadas respecto a las que tienen sombra.

Test nº2

Dado los resultados iniciales, procedemos a mejorar las propiedades del video cambiando la resolución a 1280x720 ajustando el enfoque manualmente y la iluminación/exposición, aunque el sistema de detección nos indica que esto puede afectar a la impresión dado el aumento significativo del uso de la CPU para el streaming del video. Además, añadimos una linterna para aumentar la luminosidad.

Resultados:

Podemos observar que es cierto que el streaming de 720p consume significativamente más recursos de la raspberry llevándola casi al límite del uso de cpu, aunque no parece afectar al rendimiento del reconocimiento de fallos.

	test 1 v2	test 2 v2	test 3 v2
link	link	link	link
error	0/12	8/12	11/12

Observamos que aumentando la calidad del video realmente no se observa una gran mejora en la detección, y dado el aumento de consumo lo mejor es dejar la resolución a 640x480 pero con el enfoque correcto.

6.2. Tests de rendimiento

En este apartado vamos a realizar una serie de pruebas para medir el rendimiento del sistema una vez ya está todo funcionando y 3 impresoras operativas conectadas. El objetivo es medir el rendimiento general de las partes del sistema, en concreto la Jetson y las raspberrys, para ver si tenemos alguna limitación de recursos con 3 máquinas y si no es así estimar cuántas impresoras simultáneas podría soportar.

Para ello vamos a poner en marcha 3 impresiones de prueba a la vez, acceder al portal desde varios dispositivos y medir el consumo de recursos de los dispositivos.

Jetson:

```
top - 17:50:01 up 6 days, 7 min,  2 users,  load average: 0,99, 0,75, 0,59
Tasks: 301 total,   2 running, 299 sleeping,   0 stopped,   0 zombie
%Cpu(s): 15,1 us,  1,8 sy,  0,0 ni, 79,2 id,  1,1 wa,  1,6 hi,  1,2 si,  0,0 st
KiB Mem :  4059272 total,   763120 free,  2592492 used,   703660 buff/cache
KiB Swap:  2029632 total,  1605496 free,   424136 used.  1247312 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 6694 root      20   0  156108 119980   3552 R  47,4  3,0  73:41.34 python
 9355 jetson    20   0   50808  38672   4392 S  13,8  1,0   8:51.84 uwsgi
```

Fig. 11: processos de jetson con todo en ejecución 1

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 6694 root      20   0  156108 119384   3452 S  19,8  2,9  69:29.11 python
 9355 jetson    20   0   50744  33628   4392 S   5,9  1,0   7:42.78 uwsgi
28109 jetson    20   0   10372   3748   2924 R   1,0  0,1   0:05.21 top
  836 root      -51   0       0      0      0 S   0,3  0,0   3:45.57 irq/69-host_syn
```

Fig. 12: processos de jetson con todo en ejecución 2

Como podemos observar el consumo de cpu de la jetson principalmente lo causa dos procesos, python y uwsgi [11], el coste de python se reparte entre el software de IA que detecta errores en el video y la herramienta de gestión. Pero realmente el coste de el modulo de IA es mucho más elevado que el del portal web así que podemos asumir que el consumo del proceso python es el consumo total del software de detección. Por otro lado, uwsgi es el software que hace de enlace entre el servidor, en este caso nginx y la tecnología que tiene detrás, en este caso django. En definitiva toda la información que va desde django al navegador donde se está cargando pasa por uwsgi, ya sea el video en directo de la impresión, información de esta, temperatura, etc.

En general después de analizar los recursos consumidos a lo largo de la impresión simultánea que hemos utilizado de prueba hemos observado que el consumo alcanzaba en momentos puntuales el 50 % de uso de cpu pero nunca lo supera y el consumo medio está en torno a un 35 %.

Raspberry: En el caso de la raspberry hemos detecta-

```
top - 17:04:58 up 18 days, 23:40,  2 users,  load average: 0.95, 0.96, 1.00
Tasks: 130 total,   2 running, 128 sleeping,   0 stopped,   0 zombie
%Cpu(s):  3.5 us,  2.2 sy, 14.6 ni, 79.6 id,  0.0 wa,  0.0 hi,  0.2 si,  0.0 st
MiB Mem :    873.3 total,     76.3 free,    600.1 used,    196.8 buff/cache
MiB Swap:    100.0 total,      0.2 free,     99.8 used.    170.0 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 1083 pi        30  10  110320   4804   2408 R  59.6   0.5  14832:34 ffmpeg
  403 pi        20   0 1028312 460624   6420 S   8.6  51.5   2346:33 octoprint
  710 pi        20   0  596864  48980  23404 S   2.0   5.5 590:20.57 octodash
  877 pi        30  10  136076  12552   8392 S   1.7   1.4 257:58.99 janus
  786 pi        20   0  345888  85388  40544 S   1.3   9.5 519:57.07 octodash
```

Fig. 13: Processos de raspberry en ejecución

do que el mayor consumo es la transmisión de video por red continuamente. Aunque el video está configurado a 640x480 y 10 fps supone un 60 % de la cpu de la raspberry tener que procesar y enviar imágenes a esa velocidad, los otros servicios como octoprint y derivados no suponen una carga significativa.

6.3. Tests de resultados de detección

Lo siguiente es probar parámentros como el color del filamento y el ángulo e iluminación de la cámara. Cabe destacar que el filamento utilizado a lo largo de todo el proyecto es PLA [14], un polímero barato, fácil de imprimir

y biodegradable.

Color:

Primero hemos hecho pruebas con la misma maquina, mismo ángulo y misma iluminación pero cambiando el color del filamento, para ver cuánto influye el color en la detección. Hemos observado que cada color detecta el suficiente error como para pararse en momentos diferentes, pero la diferencia entre estos creemos que no es significativa. A continuación los resultados de los colores probados, cabe destacar que aunque el porcentaje de impresión en el que se ha detenido la impresión varía, ésta tiene un tiempo total de 10 minutos, por lo tanto aunque varíe en un orden de un 10 % la diferencia en tiempo es de 1 o 2 minutos. Comentar también que el sistema se detiene automáticamente a partir de un 66 %. A.4

- Blanco, error: 66.96 % progreso impresión: 89.04 %
- Marron, error: 69.94 % progreso impresión: 92.5 %
- Naranja, error: 69.03 % progreso impresión: 84.24 %
- Negro, error: 71.47 % progreso impresión: 92.31 %

Ángulos:

Luego hemos pasado ha probar diferentes ángulos de cámara. Como norma general la cámara se encuentra anclada a una esquina de la impresora logrando un ángulo de 45º respecto al eje Y. La cámara dispone de un brazo para tener margen para enfocar diferentes zonas, más atrás, más arriba, etc. Los escenarios que hemos probado son con la cámara en esta posición (45º) y variando la cercanía, es decir con un plano más alejado y otro más cerca, y un último caso donde hemos situado la cámara en el centro. A continuación los resultados:

- 45º cerca, error: 66.96 % progreso impresión: 83.22 %
- 45º lejos, error: 67.54 % progreso impresión: 87.32 %
- centrado, error: 67.03 % progreso impresión: 97.43 %

Iluminación:

Por último pasamos a comprobar cómo afecta la iluminación. Hemos añadido una luz led a la impresora que se enciende cada vez que va a imprimir para asegurarnos que siempre estará iluminada, pero queremos probar cómo afecta esta a la IA. Para ello probaremos los siguientes escenarios: Luz apagada pero luz de ambiente (1), luz encendida y luz de ambiente (2), encendida y sin luz de ambiente(3) y apagada y con poca luz ambiente(4).

- 1, error: 66.56 % progreso impresión: 86.11 %
- 2, error: 67.14 % progreso impresión: 89.63 %
- 3, error: 67.03 % progreso impresión: 92.48 %
- 4, error: 67.03 % progreso impresión: 97.48 %

6.4. Conclusiones de los tests

Después de realizar la fase de tests podemos concluir que a priori:

- La iluminación influye en la detección, pero no significativamente.
- El ángulo más óptimo es de 45º respecto a la superficie de impresión.
- Los diferentes colores influyen en la detección, pero no significativamente.
- El consumo de cpu medio aproximado por impresora es de un 12 % de la capacidad de cómputo de una jetson.

Con este último resultado podemos inferir que nuestro sistema, utilizando una jetson nano como dispositivo central de cómputo, puede soportar hasta un máximo de 6 impresoras imprimiendo simultáneamente, sin que el rendimiento general del sistema se vea afectado. A partir de ese número no se garantiza ni la correcta detección ni que el portal web se mantenga estable.

También cabe destacar que estas conclusiones se han realizado con los resultados de pocos tests, con los que se ha podido validar que el sistema de detección es robusto y su comportamiento no depende de condiciones externas. El sistema se comportará como se espera mientras está la cámara enfocada en la impresión con un mínimo de iluminación.

Al realizar las pruebas hemos detectado que el sistema a veces detectaba como error la rueda que tiene la impresora para ajustar el nivel de la cama. Esto ha sido útil ya que podría influir en un falso positivo, aunque aún no hayamos tenido ninguno. Para solucionarlos simplemente hemos puesto un poco de cinta blanca, en un futuro se imprimirá una rueda diferente que no sea detectada.

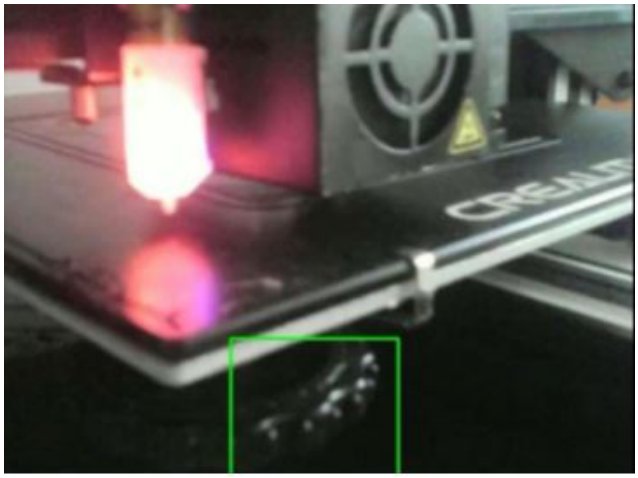


Fig. 14: Detección errónea

6.5. Estimación de coste del sistema

El coste del sistema sin tener en cuenta el precio de la impresora ronda los 110€ para una sola máquina y a partir de ahí 50€ por cada impresora añadida, ya que cada impresora necesita una raspberry pi que cuesta alrededor de 40€ y unos pocos componentes hardware.

7 Conclusiones

Podemos extraer como conclusiones que, en general, el sistema cumple con los requisitos y expectativas previstas. Se ha logrado mejorar notablemente la accesibilidad y control de hasta 3 impresoras, mediante el uso de nuestro portal web. Además, el sistema de visión por computador, después de realizar test para medir su rendimiento, ha demostrado cumplir con lo esperado en todas las medidas. Esto nos deja con un sistema capaz de vigilar las impresiones en curso y prevenir gastos de material y posibles averías en caso de que la impresión falle, sin importar demasiado aspectos como la calidad de la webcam o el tipo de color del filamento. Con nuestro sistema ya no es necesario estar pendiente de las impresiones y en el caso que se quiera no tienes que desplazarte físicamente al lugar donde se encuentra la máquina, sino que simplemente accedes a la herramienta desde cualquier lugar con conexión o acceso a la red donde se encuentre alojado el sistema.

Otra de las problemáticas más significativas que nuestro sistema soluciona es a la hora de cargar archivos en las máquinas. Antes cada vez que querías imprimir tenias que ir a buscar la memoria extraible de la máquina y cargar el archivo manualmente conectándolo al ordenador, y luego volver a la máquina y ponerla a imprimir manualmente. Ahora simplemente una vez tienes el archivo listo para imprimir solo tienes que subirlo a la memoria interna de la impresora a través de nuestro portal y acto seguido encender la máquina y ponerlo a imprimir, sin levantarte del ordenador. Esto supone una gran optimización del tiempo que usualmente se requiere dedicar para imprimir las piezas deseadas.

8 Problemáticas abiertas

Hemos detectado un error en el sistema, y es el hecho de que el streaming de video de las webcams de las impresoras, se ralentiza mucho cuando haces muchos accesos al mismo video. Es decir, cuando accedes al portal web desde varios sitios a la vez, y accedes a las cámaras, se puede llegar a sumar hasta un retraso de 10 segundos. Tras investigar posibles causas, llegamos a la conclusión de que el problema reside en el hecho de que, tal como está diseñado el sistema, es la jetson la que actúa de router para gestionar las peticiones de acceso a las webcams de cada raspberry, todo ello a través del wifi USB. Al principio creíamos que era el propio dispositivo wifi el que no tenía suficiente banda ancha para transmitir el video, pero tras analizar el consumo de la red llegamos a la conclusión que no era una limitación del hardware wifi sino que era debido a la mala gestión por parte de ubuntu de las peticiones. Para llegar a esta conclusión simulamos una parte del sistema conectado a un router normal y comprobé cuantos accesos podía soportar, y resultó que daba igual los accesos que se hiciera, el video iba a tiempo real. Para solucionar este problema, se ha decidido modificar la arquitectura de la red y añadir un acces point que gestione la red interna en vez de usar a la propia jetson. De esta manera tendríamos un acces point conectado a la red general, y será este el que gestione una red interna donde estarán conectados todos los dispositivos del sistema (raspberrys y jetson). Aún se está trabajando en esta solución para ver si es factible.

Agradecimientos

Se agradece el asesoramiento y la ayuda para la realización de este proyecto a Jordi Carrabina Aróztegui y Marc Codina Barbera por hacer el seguimiento del proyecto, por sus correcciones y puntualizaciones de este informe, y por las sugerencias y asistencias recibidas en el enfoque, tanto a nivel general, como en aspectos concretos del trabajo. Además de estar siempre dispuestos a adaptarse a mis horarios para poder reunirnos cuando era necesario. También agradecer a Marc Codina y su departamento así como a la UAB por proporcionarme una Jetson Nano de Nvidia para poder desarrollar este proyecto y hacer pruebas con el dispositivo.

Referencias

[1] Stephen Cass. 2020. Nvidia makes it easy to embed AI: The Jetson nano packs a lot of machine-learning power into DIY projects-[Hands on]. *IEEE Spectrum* 57, 7 (2020), 14–16.

[2] TheSpaghettiDetective community. 2019. marlin oficial site. `https://github.com/TheSpaghettiDetective/` [Internet; descargado 14-junio-2021].

[3] Scott Cromar. 2013. GanttProject. In *From Techie to Boss*. Springer, 225–229.

[4] Heather A Johnson. 2017. Trello. *Journal of the Medical Library Association: JMLA* 105, 2 (2017), 209.

[5] klipper community. 2018. klipper oficial site. `https://www.klipper3d.org/` [Internet; descargado 14-junio-2021].

[6] marlin community. 2016. marlin oficial site. `https://marlinfw.org/` [Internet; descargado 14-junio-2021].

[7] octoprint community. 2016. marlin oficial site. `https://octoprint.org/` [Internet; descargado 14-junio-2021].

[8] Daniel Rubio. 2017. *Beginning Django*. Springer.

[9] James Turnbull. 2014. *The Docker Book: Containerization is the new virtualization*. James Turnbull.

[10] Eben Upton and Gareth Halfacree. 2014. *Raspberry Pi user guide*. John Wiley & Sons.

[11] uWSGI community. 2019. The uWSGI proyect oficila site. `https://uwsgi-docs.readthedocs.io/en/latest/` [Internet; descargado 14-junio-2021].

[12] Wikipedia. 2019. Xming — Wikipedia, La enciclopedia libre. `https://es.wikipedia.org/w/index.php?title=Xming&oldid=119555659` [Internet; descargado 14-junio-2021].

[13] Wikipedia. 2021. PuTTY — Wikipedia, La enciclopedia libre. https://es.wikipedia.org/w/index.php?title=PuTTY&oldid=133636542 [Internet; descargado 14-junio-2021].

[14] Wikipedia. 2021. Ácido poliláctico — Wikipedia, La enciclopedia libre. https://es.wikipedia.org/w/index.php?title=%C3%81cido_polil%C3%A1ctico&oldid=133889019 [Internet; descargado 14-junio-2021].

APENDICE

A.1. Diagrama de Gantt previsto y ejecutado

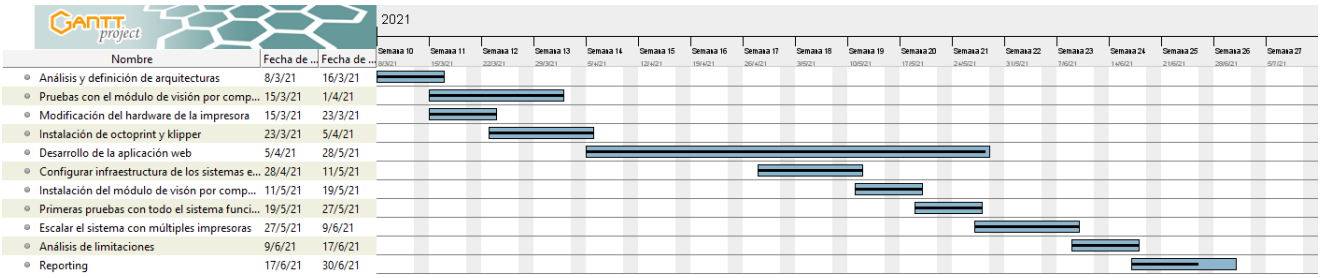


Fig. 15: Gantt previsto

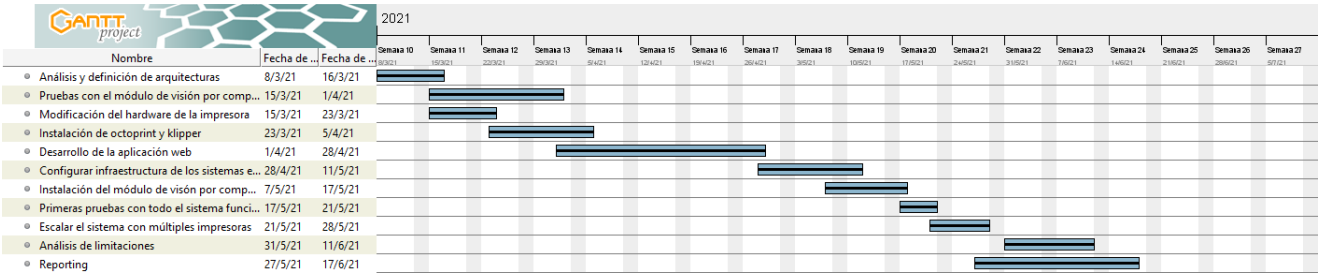


Fig. 16: Gantt ejecutado

A.2. Esquema eléctrico del sistema electronico

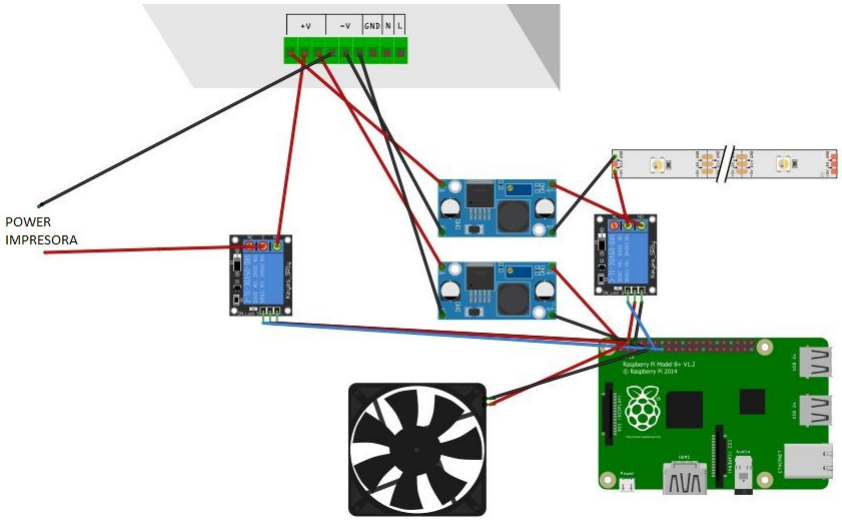


Fig. 17: diagrama electronico

A.3. Reglas de routing para configurar el servidor central(jetson)

sudo iptables -t nat -A PREROUTING -p tcp –dport 5000 -j DNAT –to-destination 10.42.0.131:80 //x each raspy
sudo iptables -t nat -A POSTROUTING -j MASQUERADE sudo iptables -I FORWARD -i eth0 -o wlan0 -j ACCEPT
sudo iptables -I FORWARD -i eth0 -o wlan0 -m state –state NEW,ESTABLISHED,RELATED -j ACCEPT
sudo iptables -D FORWARD 4 //If there is a rule opposite to the previous one, delete it

A.4. Imagenes sobre la detección con diferentes colores

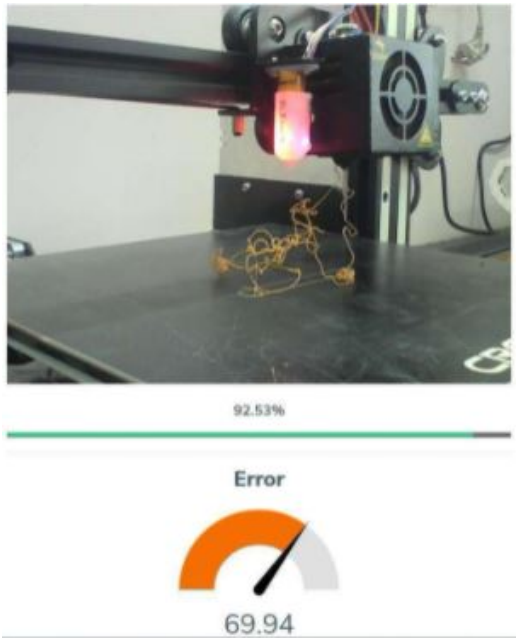


Fig. 18: detección naranja

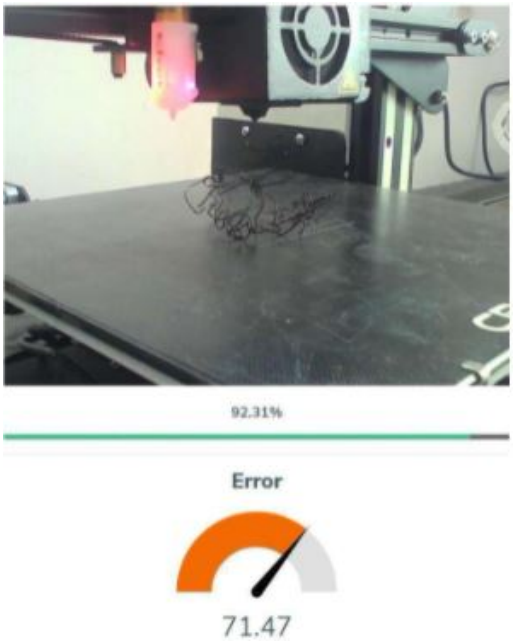


Fig. 19: detección negro

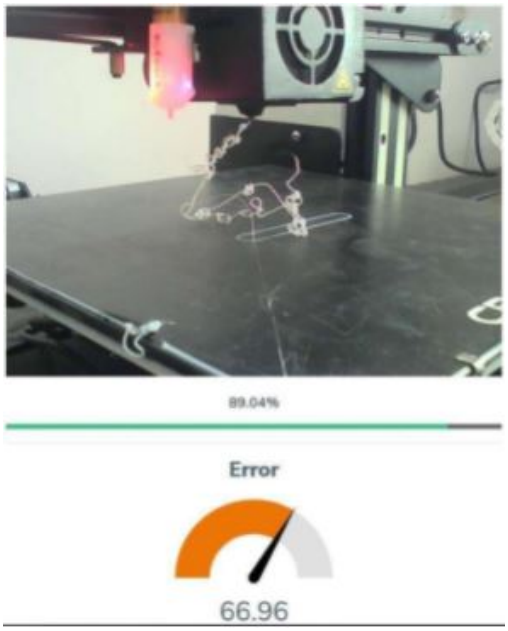


Fig. 20: detección rosa

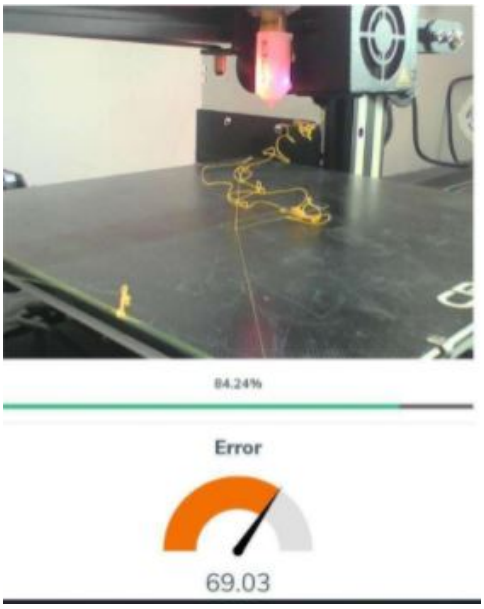


Fig. 21: detección amarillo